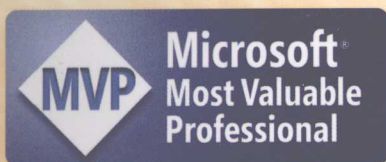


Practical .NET2 and C#2

Harness the Platform, the Language, the Framework

C# 和 .NET 2.0 实战

平台、语言与框架



[法] Patrick Smacchia 著
施 凡 李永伦 谭颖华 徐 宁 译

- C# 传奇经典
- 世界众多 .NET 专家好评如潮的秘籍
- 国内四位 MVP 联袂翻译
- 深入全面，知识密集，代码丰富



人民邮电出版社
POSTS & TELECOM PRESS



图灵程序设计丛书 .NET系列

C#和.NET 2.0实战

平台、语言与框架

Practical .NET2 and C#2
Harness the Platform, the Language, the Framework

[法] Patrick Smacchia 著

施凡 李永伦 谭颖华 徐宁 译

人民邮电出版社
北京

图书在版编目 (CIP) 数据

C#和.NET 2.0 实战: 平台、语言与框架 / (法) 斯梅切尔 (Smacchia, P.) 著; 施凡等译. —北京: 人民邮电出版社, 2008.1

(图灵程序设计丛书)

ISBN 978-7-115-16620-3

I. C… II. ①斯…②施… III. C 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字 (2007) 第 114913 号

内 容 提 要

本书是一本知识密集的 C# 技术经典图书, Microsoft .NET MVP 力作, 众多 .NET 专家口口相传的一本秘籍。全书分为三个部分, 第一部分讲述底层的 .NET 平台, 涵盖了 .NET 各方面的基础知识和工作原理; 第二部分是 C# 语言部分, 通过与 C++ 比较的方式进行讲解, 清晰易懂; 第三部分讲述 .NET Framework 中的基本类库, 内容几乎涉及 .NET 常见领域的全部知识。

本书主要面向熟悉 .NET 的编程人员, 也适合 .NET 技术的初学者阅读。

图灵程序设计丛书

C#和.NET 2.0 实战: 平台、语言与框架

- ◆ 著 [法] Patrick Smacchia
译 施 凡 李永伦 谭颖华 徐 宁
责任编辑 傅志红 王慧敏
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市海波印务有限公司印刷
新华书店总店北京发行所经销
- ◆ 开本: 800×1000 1/16
印张: 49.75
字数: 1 429 千字 2008 年 1 月第 1 版
印数: 1—4 000 册 2008 年 1 月河北第 1 次印刷
著作权合同登记号 图字: 01-2006-5780 号

ISBN 978-7-115-16620-3/TP

定价: 99.00 元

读者服务热线: (010)88593802 印装质量热线: (010)67129223

版 权 声 明

Practical .NET2 and C#2 (0-9766132-2-0) by Patrick Smacchia.

Copyright © 2005-2006 by Patrick Smacchia.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system without written permission from Paradoxal Press, except for the inclusion of brief quotations in a review.

The Paradoxal Press logo and related trade dress are trademarks of Paradoxal Press and may not be used without written permission.

Educational facilities, companies, and organizations interested in multiple copies or licensing of this book should contact the publisher for quantity discount information. Training manuals, CD-ROMs, and portions of this book can be tailored for specific needs.

本书中文简体字版由 Paradoxal Press 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

致中国读者

人们普遍认为，微软在.NET上押宝成功了。在短短5年中，.NET成为主流平台，而且.NET开发人员还在稳定地增长。随着.NET 2.0的发布，这一平台于2005年末终于成熟。现在看来，.NET的未来版本将建立在.NET 2.0核心的基础之上，而核心本身将不再变化。

在这样的大背景下，开发人员牢固地掌握.NET 2.0核心将是非常明智的。这正是本书的目的所在。在撰写本书的时候，我希望将注意力集中在一些核心的概念上，这些概念有助于读者编写出高质量的代码。.NET 2.0框架有2万多个类，乍看起来令人生畏。但是我相信，如果首先花时间了解.NET的类型系统及其与C#语法的关系，开发人员的工作很快就能变得非常高效。.NET平台本身也有核心，就是CLR（Common Language Runtime，公共语言运行库）。有些CLR的内部细节对于编写正确的代码是至关重要的，比如.NET的对象生命周期模型和内存模型。本书讲述了CLR的内部细节和C#的各种微妙之处，并介绍了.NET框架的每个部分。

希望大家获得愉快的阅读体验！

Patrick Smacchia

2007年10月

To Chinese Readers

It is now widely accepted that Microsoft succeeded its bet on .NET. In half a decade .NET became a mainstream platform and the .NET developers community is constantly growing. The platform became mature at the end of 2005 with the release of .NET 2.0 . We now see that further version of .NET are based on the .NET 2.0 core that won't change anymore.

In this context, it is judicious for developers to master the .NET 2.0 core and this is what this book is all about. While writing *Practical .NET2 and C#2*, I wanted to make sure to focus on core concepts that will help readers write quality code. With more than 20,000 classes, the .NET 2.0 framework is daunting at first glance. However, I believe that developers can quickly become productive if they first take the time to learn about the .NET type system and how it relates to C# syntax. Also the .NET platform has a heart, the CLR (Common Language Runtime). There is some CLR internals that are essential to write correct code, such as the .NET objects lifecycle model and the memory model. The book covers CLR internals and the C# subtleties, but also introduces each part of the .NET framework.

Enjoy reading!

Patrick Smacchia

2007, 10

对本书的赞誉

“Patrick写就了一本令人称奇的C#图书!”

——Scott Guthrie, ASP.NET之父, 微软开发事业部总经理

“本书涵盖的知识面之广之深, 给我留下了深刻印象。读每一章我都能学到很多新东西和实用的技巧。”

——Mike Stall, 微软CLR小组成员

“我的桌上曾经有半打C#图书, 现在我想它们都可以被本书取代了。”

——Steven Smith, 微软区域经理

微软ASP.NET MVP, AspAlliance.com和AspAdvice.com总裁

“书非常棒, 充满了细节。阅读本书是一种乐趣。”

——David Hayden, 微软C# MVP兼网站开发专家

“如果你不想在5本书中来回寻找某个技术细节, 请把本书放在手边。”

——Sahil Malik, 《ADO.NET高级程序设计》作者, 微软C# MVP

“.NET平台中所涉及的内容基本上都能在本书中找到。”

——Fritz Onion, *Essential ASP.NET*作者, 微软MVP

本书应该放在电脑桌上, 随时翻阅。

——Peter Bromberg, 微软C# MVP, EggHeadCafe.com站长

本书让我印象最深刻的地方在于, 它很好地兼顾了内容的广度与深度。

——Mike Taulty, 微软英国公司开发人员技术推广官

我向所有想扎实掌握.NET框架的开发人员推荐本书。

——John Papa, 微软C# MVP, 多本.NET技术图书的作者

本书绝对应该拥有!

——Gabriel Lozano-Mor, 独立咨询师, MCSD.NET

2 对本书的赞誉

我发现自己现在不依靠这本书，经常无法完成手头的工作。

——Bill Ryan，微软Windows Embedded MVP，*Professional ADO.NET 2* 作者之一

这一部瑞士军刀式的C#和.NET图书，我强烈向所有.NET开发人员推荐。

——Miha Markic，微软C# MVP

本书对于编写.NET 2.0应用程序的初中级开发人员而言，是一部非常好的书。

——Mahesh Chand，微软MVP，www.c-sharpcorner.com站长，
Graphics Programming with GDI+ 一书作者

本书非常全面……真是一部优秀的C#著作。

——Wesner Moise，前微软程序员，博客Smart Software（曾经名为.NET Undocumented）作者

Patrick阐明了大量有趣的技术细节。

——Mike Gunderloy，Larkware.com 站长，*Coder to Developer*一书作者

译者序

本书全面介绍了.NET Framework 2.0和C# 2.0。作者Patrick Smacchia是来自法国的微软MVP，他在书中展现了丰富的经验和卓越的逻辑性。本书不但涉及的知识面宽，而且每个主题都具有相当的深度，可以让读者在“知其然”的基础上“知其所以然”。因此它不但适合.NET技术的初学者，熟悉.NET的程序员也可以从中获得有价值的信息。本书内容的深度和广度足以使其获得“.NET面试宝典”的称号，正在准备.NET技术面试的朋友不妨参考一下。

本书分为三个部分。第一部分主要介绍.NET Framework的基础，包括程序集、IL、CLR、线程、反射和安全性等.NET架构的基础设施。这一部分涵盖了.NET各方面的基础知识和工作原理。而且对.NET 2.0新增的功能（如ClickOnce等）也有相应的介绍。第二部分讲解C#语言。作者采用了对C#和C++进行比较的方式讲解，因此熟悉C++的读者可以快速了解C#与C++的异同。不懂C++的读者也可以直接看C#部分，因为两种描述是相互独立的。这一部分不但介绍了C#语言的基础语句、类型系统、面向对象等特性，还特别讲解了C# 2.0新增的泛型、匿名方法和迭代器等特性。第三部分介绍的是.NET Framework中的类库。这一部分的内容十分丰富，几乎涉及从.NET集合、基本数学类库到Windows Forms、.NET Remoting、ASP.NET等.NET常见领域的全部知识。特别是第23章详细介绍了ASP.NET的完整功能，非常有参考价值。

本书各章的设计注重独立性，因此读者可以不必从头看到尾，而可以随意挑选任意章或按照自己喜欢的顺序来进行阅读。书中一大特点是章节之间的相互引用非常丰富，有助于读者了解一种技术的所有相关信息。我们建议不熟悉.NET的读者先阅读第二部分，因为获得C#语法的相关知识后就更有利于理解第一部分的代码。书中的示例程序也是本书的一大特色，不仅数量丰富，而且所有的示例程序均可编译运行。

本书前言、第1章、第2章、第4章、第9章~第14章以及第8章的部分小节由施凡翻译，第3章、第5章~第7章和第24章以及第8章后半部分由徐宁翻译，第15章~第18章、第21章~第22章以及第8章前半部分由李永伦翻译，第19章~第20章和第23章由谭颖华翻译。术语的翻译大多参考MSDN中文版中使用的译法，当MSDN不准确的时候又参考了流行的约定译法以及部分学术文献，力求准确。有些译法暂时未获得广泛认可的术语，我们保留了原文，比如.NET attribute。我们希望本书能够比较准确、通畅地表达作者想要表达的所有想法。但受限于译者的水平，本书难免存在技术上的漏洞与表达错误的问题。恳请读者对我们的工作提出批评与建议。

在翻译本书的过程中得到了博客园 (<http://www.cnblogs.com>) 技术博客上多位网友的协助，在此对他们一并表示感谢。衷心希望读者能够喜欢本书。

译者

2007年于微软亚洲工程院

译者简介

施凡 曾3次获得微软MVP称号,现工作于微软中国研发集团MBDC部门。从2001年开始接触.NET开发,是VB.NET方面的专家,在CLR、语言编译器和泛型编程方面亦有很高的兴趣。平时喜欢阅读技术Blog和编程书籍。

李永伦 自2006年4月连获两届微软MVP,现就职于高知特信息科技有限公司(上海)有限公司,管理科班生,专职程序员,业余咨询师。工作之余喜欢阅读《新发现》、《经理人》和《心理月刊》,与同道中人讨论新技术、软件设计、开发方法学和企业管理,以及为有需要的人提供技术培训。好于博客中长篇大论自己之见解,却醉于阅读他人短小精巧之言论。你可以在<http://www.cnblogs.com/allenlooplee>阅读到他的技术文章,在<http://www.manageblog.net/allenlooplee>接触到他的管理文章,也可以通过allenlooplee@yahoo.com.cn与他交流。

谭颖华 2006年4月获得微软MVP。现任广州图睿信息技术有限公司运营总监,有多年的大型系统开发和实施经验,曾任职于汇丰银行广州软件研发中心。

徐宁 2007年7月获得微软C# MVP。对软件设计开发以及分布式技术有较大兴趣。现工作于道富信息科技有限公司(浙江)有限公司,平时喜欢阅读技术博客和编程书籍,在博客园发表多篇技术文章。你可以在<http://idior.cnblogs.com/>阅读到他的技术文章,也可以通过xuning.net@gmail.com与他联系。

前言

关于本书

微软公司的.NET MSDN文档十分庞大，详细讲述了数千种类型。MSDN文档还包含了许多关于如何使用.NET各部分的文章。作为一名开发人员，我十分清楚在使用微软技术进行开发时MSDN文档的重要性。然而，MSDN的篇幅过于庞大，很难从整体的角度把握.NET提供的功能。从我的经验来看，新的想法和新的概念还是从书本中获得更好。当然，你可以把MSDN的上万页文档打印到纸上，但想要将其带到花园或者躺椅上静静地阅读就很困难了。

本书的初衷是与MSDN结合起来使用。所以其目的不是要把数千个.NET类型的数千成员都列举一遍，而是用简明且有用的例子来解释和展示.NET平台、C#语言以及.NET Framework的诸多方面。我希望本书能够帮助读者了解技术背后的设计动机，并能够带领读者探索现代软件开发的全新途径。

本书结构

第一部分：.NET 平台

本书第一部分描述的是.NET平台的底层架构。它回答诸如下面这些问题：

- .NET应用程序的执行与底层的操作系统有什么联系？
- 编译器将应用程序编译之后生成的文件结构是什么样的？
- 资源访问以及安全性是如何管理的？
- 如何才能利用这些因素提高应用程序的质量和性能？
- 如何才能.NET应用程序中利用以前在Windows下开发的代码？

第二部分：C#语言以及 C# 2.0 与 C++的比较

第二部分全面讲述C# 2.0语言。C#比C++更接近于Java。因此本书从C#语言的各个方面尽力描述了C++和C#的异同，希望这样能够快速解答所有从C++转向C#的开发人员可能会有的疑问。

第三部分：.NET Framework

第三部分描述.NET Framework的基类。这些类的功能可以分为以下几类：

- 集合。
- 用于完成数学运算、日期和时间、文件夹和文件、跟踪和调试、正则表达式及控制台的经典基类。
- 使用数据流管理I/O。
- 使用Windows Forms 2.0开发图形应用程序。
- 使用ADO.NET 2.0管理数据库。

- 事务管理。
- 创建以及处理XML文档。
- 使用.NET Remoting创建分布式对象的应用程序。
- 使用ASP.NET 2.0开发Web应用程序。
- Web服务。

关于本书结构的说明

本书这样安排是为了让读者在阅读中随时清楚自己读到了哪一部分。显然如此庞大的技术在许多方面都会超越本书的结构。比如,我们选择在第一部分讲解如何对资源的并发访问进行同步,因为这一内容基于底层的线程和进程的概念,而这些概念又与系统平台相关。然而用于同步的类也是.NET Framework的一部分,因而也可以放在第三部分介绍。最后,C#语言包含了一些特殊的关键字来简化这些类的使用,因此这个主题也会在第二部分有所涉及。

本书包含了很多交叉引用,希望这些引用能够引导读者在各个不同的主题间自由转换。

本书的读者对象

只要你对在.NET下进行开发感兴趣,就是本书的读者,无论你是学生、专业的或业余的开发人员、教师、架构师还是技术团队的领导。

本书每一章的设计都适于顺序阅读,但整本书却不是这样。第一部分.NET平台是最难理解也是最基本的一部分(而且本人认为是最有趣的一部分)。如果不能很好地理解底层的平台,显然就不可能很好地用.NET来进行开发。

初学者可以从C#语言和对象开发技术学起,一步步掌握.NET。

熟悉.NET以外其他技术的读者可以从.NET诸多创新性的特性的讲解中获益。

熟悉.NET 1.x的读者可以查看附录B,它涵盖了书中介绍的.NET 2.0的所有新特性。

支持

本书英文版的配套网站为<http://www.PracticalDOT.Net/>。

从这个网站可以下载本书的所有示例。我们深信,良好的例子比冗长的讨论更有价值。本书共有647个例子,其中523个是用C#编写的,有100多个是专门用于ASP.NET 2.0的。所有这些例子都可以在<http://www.PracticalDOT.Net/>找到。

致谢

首先我要感谢我的朋友Eli Anc,她在我撰写本书的过程中给了我莫大的支持。我还要郑重感谢Francis、France、Michel、Christine、Mathieu、Julien、Andrée、Patrick、Marie-Laure和Philippe的支持。

感谢Paradoxal出版社的Sébastien St Laurent,感谢他卓越的专业知识和对本书翻译及出版所作的贡献。同样感谢O'Reilly法国公司的Xavier Cazin,他从.NET刚出现就一直支持并帮助我编写本书。

我还要感谢所有帮助审校本书以及对本书提出有价值的建议的朋友们:

Alain Metge, 18年专业经验。负责法国南部高速公路软件的架构设计。

Bertrand Le Roy博士, 8年专业经验,参与了微软公司的ASP.NET技术的构想达3年之久。

Bruno Boucard, 18年专业经验,任法国兴业银行架构师及培训师8年。微软资深架构师。

Frédéric De Lène Mirouze (即Améthyste), Web开发专家, 20年专业经验,曾与ELF、Glaxo、Nortel、Usinor等多家公司合作。获MCAD.NET认证。

Jean-Baptiste Evain, 3年专业经验, 公共语言基础设施 (CLI) 方面的专家, 参与了Mono与AspectDNG。

Laurent Desmons, 10年专业经验, 架构师及.NET技术顾问, 曾与Péchiney、Arcelor、Sollac等公司合作。获MCSD.NET认证。

Matthieu Guyonnet-Duluc, 4年专业经验, 法国电信商业软件开发人员。

Michel Futersack博士, 巴黎第五大学计算机专业教师, 在OOP概念和编程方面有10多年的教学经验。

Nicolas Frelat, 具有4年实践经验的.NET技术顾问。.NET 2.0平台的早期采用者。

Olivier Girard, 6年专业经验, EAI专家, 法兰西银行的架构师。

Patrick Philippot, 自由撰稿人, 30年专业经验 (在IBM工作19年), .NET的MVP, www.mainsoft.fr 站长。

Sami Jaber, 8年专业经验, Valtech的培训师及高级顾问, 曾与空中客车公司合作, www.dotnetguru.org 的站长。

Sébastien Ros, 7年专业经验, 对象-关系映射专家, DTM工具的设计者, Evaluant (www.evaluant.com) 的CTO。

Sébastien Vaucouleur, 8年专业经验, 语言专家, 曾与Bull和富士通合作, 瑞士苏黎士ETH大学研究助理。

Thibaut Barrère, 自由撰稿人, J2EE/.NET/C++平台专家, 从1984年开始编程, 最近曾与Calyon、PPR/Redoute和MCS合作。参与开发CruiseControl.Net、NAnt与TestDriven.Net等开源软件。

Thomas Gil, 8年专业经验, 面向方面编程 (AOP) 专家, AspectDNG项目主管, 顾问及独立培训师, www.dotnetguru.org 副站长。

Vincent Canestrier, 曾任教于法国国立工艺学院, 安永公司前技术总监。

我还想感谢微软工程师Brian Grunkemeyer、Florin Lazar、Krzysztof Cwalina和Michael Maruchek, 感谢他们对我所提问题提供了宝贵的答案。

最后, 我想感谢所有选择本书的朋友。我真心希望本书能够帮助你完成所有的任务。

Patrick Smacchia

目 录

第 1 章 .NET 简介	1
1.1 什么是.NET	1
1.1.1 微软软件开发平台	1
1.1.2 一组规范	1
1.1.3 .NET 概览	1
1.2 发展历程	2
1.2.1 过去	2
1.2.2 现在	2
1.2.3 未来	3
1.3 微软和 Windows 以外的.NET	3
1.3.1 ECMA 组织与.NET	3
1.3.2 W3C 联盟	4
1.3.3 Mono 项目	4
1.3.4 微软 SSCLI 项目	4
1.4 .NET 资源链接	5
1.4.1 网站	5
1.4.2 新闻组	5
1.4.3 博客	6

第一部分 .NET 平台

第 2 章 程序集、模块和 IL 语言	10
2.1 程序集、模块和资源文件	10
2.1.1 程序集和模块	10
2.1.2 资源文件	10
2.1.3 程序集、模块、类型和资源	10
2.1.4 为何对多模块程序集感兴趣	11
2.1.5 ILMerge 工具	11
2.2 模块的剖析	11
2.2.1 可移植的可执行文件简介	11
2.2.2 模块的结构	11
2.2.3 清单的结构	12
2.2.4 类型元数据段的结构	12
2.3 使用 ildasm.exe 和 Reflector 工具 分析程序集	14
2.3.1 创建需要分析的程序集	14

2.3.2 使用 ildasm.exe 分析模块	15
2.3.3 Reflector 工具	16
2.4 程序集 attribute 和版本设定	17
2.4.1 程序集的标准 attribute	17
2.4.2 程序集的版本设定	18
2.4.3 友元程序集	18
2.5 强名称程序集	19
2.5.1 简介	19
2.5.2 sn.exe 工具	19
2.5.3 公钥记号	20
2.5.4 为程序集签名	21
2.5.5 具体示例	21
2.5.6 程序集的延迟签名	23
2.6 国际化/本地化与卫星程序集	23
2.6.1 区域设置和本地化	23
2.6.2 资源文件	24
2.6.3 在代码中使用资源	25
2.6.4 创建卫星程序集	26
2.6.5 部署和使用卫星程序集	26
2.6.6 避免在资源无法找到时引发 异常	27
2.6.7 Visual Studio 与卫星程序集	28
2.6.8 区域设置与字符串格式化	28
2.7 IL 语言简介	29
2.7.1 栈及其特殊的 IL 指令	29
2.7.2 示例 1: 局部变量与栈	29
2.7.3 示例 2: 方法调用与栈	30
2.7.4 用于比较、分支和跳转的 IL 指令	31
2.7.5 IL 的面向对象特性	32
2.7.6 元数据符号	32
第 3 章 生成、部署以及配置.NET 应用程序	34
3.1 用 MSBuild 生成应用程序	34
3.2 MSBuild: 目标、任务、属性、项与 条件	34

3.2.1	.proj 文件、目标与任务	34
3.2.2	属性	36
3.2.3	项	36
3.2.4	条件	37
3.3	高级 MSBuild	38
3.3.1	增量生成与目标间的依赖	38
3.3.2	MSBuild 转换	38
3.3.3	将一个 MSBuild 项目分解到多个文件	39
3.3.4	Visual Studio 2005 如何利用 MSBuild	39
3.3.5	创建自定义 MSBuild 任务	39
3.4	配置文件	41
3.4.1	machine.config 文件	41
3.4.2	标准配置参数	41
3.4.3	使用<appSettings>元素定义配置参数	42
3.4.4	使用配置节定义配置参数	42
3.4.5	使用 Visual Studio 2005 创建配置节	44
3.4.6	配置节的注意事项	44
3.5	程序集部署: XCopy 与 GAC	45
3.5.1	XCopy 部署	45
3.5.2	共享程序集与 GAC 文件夹	45
3.5.3	GAC 的并存存储模型是如何解决 DLL hell 问题的	46
3.5.4	并存执行	46
3.5.5	查看及编辑 GAC 文件夹	46
3.6	发布者策略程序集	47
3.6.1	潜在的问题	47
3.6.2	解决方案	47
3.6.3	创建发布者策略程序集	48
3.7	.NET 应用程序部署简介	49
3.7.1	MSI、cab、XCopy、ClickOnce 和 NTD 之间的对比	49
3.7.2	MSI 与 ClickOnce 的对比	50
3.8	使用 cab 文件部署应用程序	50
3.9	使用 MSI 技术部署应用程序	52
3.9.1	添加文件	52
3.9.2	安装快捷方式	52
3.9.3	在 GAC 文件夹中添加一个共享程序集	52

3.9.4	安装项目属性	53
3.9.5	更新注册表	53
3.9.6	指定在安装期间执行的自定义动作	53
3.9.7	为安装提供一个自定义用户界面	54
3.10	使用 ClickOnce 技术部署应用程序	54
3.10.1	部署文件夹	54
3.10.2	为 ClickOnce 部署做准备	55
3.10.3	ClickOnce 部署与移动代码安全	56
3.10.4	按需安装与下载组	57
3.10.5	更新一个使用 ClickOnce 安装的应用程序	58
3.10.6	应用程序所需的 CAS 权限集的工作机制	58
3.10.7	ClickOnce 应用程序安装与执行的细节	59
3.11	使用无接触部署 (NTD) 技术部署应用程序	60
3.12	如果目标机器上没有安装 .NET 运行库怎么办	61
第 4 章	CLR	62
4.1	应用程序域	62
4.1.1	简介	62
4.1.2	线程与 AppDomain	62
4.1.3	卸载 AppDomain	63
4.1.4	AppDomain 和孤立性	63
4.1.5	System.AppDomain 类	63
4.1.6	在一个进程中承载多个应用程序	63
4.1.7	在其他 AppDomain 的上下文中运行代码	65
4.1.8	AppDomain 类的事件	65
4.1.9	在同一个进程的 AppDomain 之间共享信息	66
4.2	在 Windows 进程内通过运行库宿主加载 CLR	67
4.2.1	mscorlib.dll 和 mscorwks.dll	67
4.2.2	mscorlib.dll 程序集	67
4.2.3	运行库宿主介绍	67

4.2.4	在同一台计算机上承载多个版本的 CLR	68	4.8	提高代码可靠性的机制	89
4.2.5	使用 CorBindToRuntimeEx() 函数加载 CLR	68	4.8.1	异步异常及托管代码可靠性	89
4.2.6	创建一个自定义的运行库宿主	69	4.8.2	受约束执行区域	90
4.2.7	在自定义运行库宿主中调整 CLR	71	4.8.3	如何定义 CER	90
4.2.8	SQL Server 2005 运行库宿主的 特性	71	4.8.4	内存门	90
4.3	剖析.NET 应用程序的执行状况	73	4.8.5	可靠性契约	91
4.4	定位和加载程序集	73	4.8.6	关键终结器	91
4.4.1	CLR 何时尝试定位程序集	74	4.8.7	临界区	92
4.4.2	CLR 使用的定位算法	74	4.9	CLI 和 CLS	92
4.4.3	配置文件的<assemblyBinding> 元素	75	4.9.1	.NET 语言必须满足的要求	92
4.4.4	定位算法示意图	76	4.9.2	从开发人员的观点看 CLI 和 CLS	93
4.4.5	影子复制机制	76	第 5 章	进程、线程与同步	94
4.5	运行库类型解析	77	5.1	简介	94
4.5.1	显式或隐式加载程序集	77	5.2	进程	94
4.5.2	编译时引用程序集	78	5.2.1	简介	94
4.5.3	示例	78	5.2.2	System.Diagnostics.Process 类	95
4.5.4	类型解析算法示意图	79	5.2.3	创建和销毁子进程	95
4.6	JIT (即时) 编译	79	5.2.4	避免在一台机器上同时运行同一 应用程序的多个实例	95
4.6.1	可移植的二进制代码	79	5.2.5	终止当前进程	96
4.6.2	即时编译技术简介	80	5.3	线程	96
4.6.3	ngen.exe 工具	81	5.3.1	简介	96
4.6.4	性能计数器与 JIT 编译	81	5.3.2	受托管的线程与 Windows 线程	97
4.7	垃圾收集器和托管堆	83	5.3.3	抢占式多任务处理	97
4.7.1	垃圾收集技术简介	83	5.3.4	进程与线程的优先级	97
4.7.2	垃圾收集算法遇到的问题	83	5.3.5	System.Threading.Thread 类	98
4.7.3	.NET 的 GC	84	5.3.6	创建与联结线程	99
4.7.4	第一步: 寻找根对象	84	5.3.7	挂起线程	99
4.7.5	第二步: 建立活动对象树	84	5.3.8	终止线程	100
4.7.6	第三步: 解除分配非活动对象	84	5.3.9	前台线程与后台线程	100
4.7.7	第四步: 清理堆碎片	85	5.3.10	受托管线程的状态图	101
4.7.8	第五步: 重新计算托管引用所 使用的物理地址	85	5.4	访问资源同步简介	101
4.7.9	推荐做法	85	5.4.1	竞态条件	101
4.7.10	针对大对象的特殊堆	86	5.4.2	死锁	102
4.7.11	多线程环境下的垃圾收集	86	5.5	使用 volatile 字段与 Interlocked 类 实现同步	102
4.7.12	弱引用	86	5.5.1	volatile 字段	102
4.7.13	使用 System.GC 类影响 GC 的 行为	87	5.5.2	System.Threading.Inter- locked 类	103
			5.5.3	Interlocked 类提供的其他 功能	104

5.6 使用 <code>System.Threading.Monitor</code> 类与 C# 的 <code>lock</code> 关键字实现同步	104
5.6.1 <code>Enter()</code> 方法和 <code>Exit()</code> 方法	104
5.6.2 C# 的 <code>lock</code> 关键字	105
5.6.3 <code>SyncRoot</code> 模式	106
5.6.4 线程安全类	106
5.6.5 <code>Monitor.TryEnter()</code> 方法	107
5.6.6 <code>Monitor</code> 类的 <code>Wait()</code> 方法、 <code>Pulse()</code> 方法以及 <code>PulseAll()</code> 方法	107
5.7 使用 Win32 对象同步: 互斥体、事件 与信号量	109
5.7.1 共享 Win32 同步对象	109
5.7.2 互斥体	110
5.7.3 事件	110
5.7.4 信号量	112
5.8 利用 <code>System.Threading.Reader- WriterLock</code> 类实现同步	113
5.9 利用 <code>System.Runtime.Remoting. Contexts.SynchronizationAttr- ibute</code> 实现同步	114
5.9.1 同步域简介	115
5.9.2 <code>System.Runtime.Remoting. Contexts.Synchronization</code> 与同步域	116
5.9.3 重入与同步域	117
5.9.4 另一个名为 <code>Synchronization</code> 的 attribute	119
5.10 CLR 的线程池	119
5.10.1 简介	119
5.10.2 使用线程池	120
5.11 定时器	121
5.11.1 <code>System.Timers.Timer</code> 类	121
5.11.2 <code>System.Threading.Timer</code> 类	122
5.11.3 <code>System.Windows.Forms.Timer</code> 类	123
5.12 异步方法调用	123
5.12.1 异步委托	123
5.12.2 回调方法	124
5.12.3 向回调方法传递状态	125
5.12.4 one-way 调用	126
5.13 线程-资源亲缘性	126
5.13.1 <code>System.ThreadStatic- Attribute</code>	127
5.13.2 线程本地存储 (TLS)	127
5.13.3 <code>System.ComponentModel. ISynchronizeInvoke</code> 接口	130
5.14 执行上下文简介	130
第 6 章 安全性	133
6.1 代码访问安全性 (CAS) 概述	133
6.1.1 什么是移动代码	133
6.1.2 CAS: 全局观	133
6.1.3 给程序集代码授予权限	134
6.1.4 在运行程序集的代码时检查 权限	134
6.2 CAS: 证据和权限	134
6.2.1 什么是证据	134
6.2.2 .NET Framework 所提供的标准 证据	134
6.2.3 谁提供证据	137
6.2.4 权限	137
6.2.5 标准权限	137
6.2.6 标识权限	138
6.2.7 安全权限	138
6.2.8 自定义权限	138
6.3 CAS: 通过应用安全策略根据证据 授予权限	139
6.3.1 安全策略级别	139
6.3.2 剖析安全策略	139
6.3.3 用于应用安全策略的算法	139
6.3.4 默认安全策略配置	140
6.3.5 配置安全策略	141
6.4 CAS: <code>FullTrust</code> 权限	142
6.5 CAS: 从源代码进行命令式的权限 检查	142
6.5.1 <code>CodeAccessPermissions</code> 类和 <code>PermissionSet</code> 类	143
6.5.2 <code>Demand()</code> 方法	143
6.5.3 <code>Deny()</code> 方法、 <code>RevertDeny()</code> 方 法、 <code>PermitOnly()</code> 方法和 <code>Reve- rtPermitOnly()</code> 方法	143
6.5.4 <code>Assert()</code> 方法和 <code>RevertAsse- rt()</code> 方法	144
6.5.5 <code>FromXml()</code> 方法和 <code>ToXml()</code> 方法	145

6.5.6	System.Security.IPermission 接口	145	6.14.2	System.Security.Crypto- graphy.ProtectedData 类	163
6.6	CAS: 使用 attribute 进行声明式的 权限检查	145	6.14.3	System.Security.Crypto- graphy.ProtectedMemory 类	164
6.6.1	加载程序集时调整所授权限 集合的 attribute	146	6.14.4	System.Security. SecureString 类	164
6.6.2	命令式与声明式的对比	146	6.14.5	保护配置文件中的数据	165
6.7	CAS: 测试和调试移动代码的实用 机制	147	6.14.6	确保网络传输中数据的安全	166
6.8	CAS: 独立存储区权限	147	6.15	使用 Authenticode 技术与 X.509 证书 验证程序集	166
6.9	.NET、Windows 用户与角色	147	6.15.1	Authenticode 与强名称	166
6.9.1	Windows 安全简介	148	6.15.2	证书与证书认证中心	167
6.9.2	IIIdentity 接口与 IPrincipal 接口	149	6.15.3	根证书	167
6.9.3	Windows 安全标识符	149	6.15.4	Windows、.NET 与 Authenti- code 技术	167
6.9.4	在底层 Windows 线程中模拟 用户	150	第 7 章	反射、后期绑定与 attribute	168
6.10	.NET 与 Windows 资源的访问控制	151	7.1	反射	168
6.10.1	Windows 访问控制简介	151	7.1.1	何时需要反射	168
6.10.2	在.NET 代码中使用特殊的 SD	152	7.1.2	.NET 反射有何新意	168
6.10.3	在.NET 代码中使用通用的 SD	154	7.1.3	对载入 AppDomain 的程序集的 反射	170
6.11	.NET 与角色	155	7.1.4	从元数据获取信息	170
6.11.1	定义应用程序域的主体策略	155	7.2	后期绑定	171
6.11.2	检查用户是否属于某个特定 角色	156	7.2.1	“绑定类”的含义	171
6.11.3	COM+ 角色	157	7.2.2	早期绑定与动态绑定	172
6.12	.NET 与密码学: 对称算法	158	7.2.3	后期绑定	172
6.12.1	对称算法概述	158	7.2.4	在 C# 编译到 IL 期间如何实例 化一个未知的类	173
6.12.2	.NET Framework 与对称 算法	158	7.2.5	使用后期绑定	176
6.13	.NET 与密码学: 非对称算法 (公钥/ 私钥)	160	7.2.6	利用接口: 使用后期绑定的 正确方法	178
6.13.1	非对称算法概述	160	7.3	attribute	180
6.13.2	安全会话简介	161	7.3.1	attribute 是什么	180
6.13.3	RSA 算法	161	7.3.2	何时需要 attribute	180
6.13.4	非对称算法与数字签名	161	7.3.3	关于 attribute 应该知道的事	180
6.13.5	.NET Framework 与 RSA 算法	162	7.3.4	可以应用 attribute 的代码元素	181
6.14	数据保护 API	163	7.3.5	.NET Framework 中的一些标准 attribute	181
6.14.1	Windows 的数据保护 API	163	7.3.6	自定义的 attribute 的示例	182
			7.3.7	条件 attribute	185
			7.4	动态生成程序集并在运行中使用	185
			7.4.1	为什么要考虑动态生成程序集	185